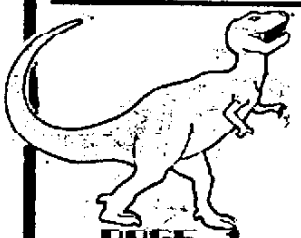
LONG ISLAND
99ER USERS GROUP

VOL #12 NO.08 AUGUST 1993 $2.00

NO MEETING AUG (TONY'S BASEMENT IS BEING REBUILT.)



LONG ISLAND SOUND

EDITOR: FRANCIS J. BUBENIK JR.

PAGE 1.....TABLE OF CONTENTS.

PAGE 2.....1993-94 COMPUTOR FAIR SCHEDULE.
COMPILED BY FRANK J. BUBENIK JR.

PAGE 3.....XB MISCELLANEOUS #26.
By EARL RAGUSE / UGOC U.G.

PAGE 4.....XB MISCELLANEOUS #26 (CONTINUED).
XB SPEED TESTS

PAGE 5.....XB MISCELLANEOUS #27.
By EARL RAGUSE / UGOC U.G.

PAGE 6.....XB MISCELLANEOUS #27 (CONTINUED).
SUB PROGRAMS - ZIGGY, PAK, GKEY.

PAGE 7.....TIPS #69 TIGERCUB SOFTWARE.
By JIM PETERSON - TIGERCUB.

PAGE 8.....TIPS #69 (CONTINUED).
SAMPLE XB PROGRAMS TO KEY IN.

PAGE 9.....NEW LITI-99ERS MEETING LOCATION.
DIRECTIONS AND MAP TO TONY'S HOME.

PAGE 10.....VERBAL VCR REMOTE. MAILER / LOGO

> HAVE A SAFE SUMMER <

ESTABLISHED APRIL-1983

COME TO THE FAIRS

* 93/94 FAIR SCHEDULE *

Compiled by FranK J. BubeniK Jr. (NL Editor)

AUG 14, 1993 (SAT) FAIRLIEIGH DICKINSON UNIV. RTE 4 ON HACHENSACK AVE. HACKENSCK,NJ 10-3PM 500 TABLES.

AUG 28, 1993 (SAT) EDISON NJ. RARITAN CENTER EXP HALL 1200 TABLES. 10-3PM. EXIT#10 NJ TNPK. KGP.

SEP 18, 1993 (SAT) WAYNE,NJ. WM PATERSON COLLEGE. 400 TABLES. RTES 46/23. 10-4PM. KGP.

OCT 30/31, 1993 (SAT/SUN) FAIRLEIGH DICKINSON UNIV. 500 TABLES.10-3PM.RTE 4 TO HACKENSACK AVE.SOUTH.KGP.

TI NOV ??&??, 1993 CHICAGO (FRI/SAT) *11th ANNUAL INTERNATIONAL FAIRE.

NOV 13/14, 1993 (SAT/SUN) EDISON, NJ. RARITAN CENTER 1200 TABLES. 10-3PM. EXIT #10 NJ TNPK. KGP.

TI NOV ??, 1993 (SUN) MILWAUKEE TI FAIRE. INFO CALL GENE HITZ (414) 535-0133.

DEC 4, 1993 (SAT) WAYNE,NJ. WM. PATERSON COLLEGE. 400 TABLES. 10-4PM. RTES. 46/23. INDOORS. KGP.

1994

TI FEB 19-20, 1994 (SAT/SUN) **FEST-WEST** TUCSON, AZ. SANTA RITA INN. INFO:CONTACT B.J.MATHIS (602) 747-5046 or TOM WILS (602) 886-2460. SOUTH WEST 99ERS U.G.

LITI 99ERS NEWSLETTER IS NOT RESPONCIBLE FOR CANCEL-ATIONS. CALL THE NUMBERS BELOW TO VERIFY TIME AND DATES BEFORE YOU GO.

* Ken Gordon Productions (KGP) SHOWS COST $8.00- $1.00 discount cards are sent to those people on there mailing lists. Call (800)631-0062 OR (908)297-2526 for info. (rev 5/15/93).

* Tri-State Computer Fairs (TSCF) SHOWS COST $6.00- $1.00 discount cards available by mail. Call Robert Barlow (201) 533-1991 for info. (rev 6/11/93).

- Don't miss the bargains on computers, software, IC's, peripherals, printers, monitors, ports, supplies and books.

XB MISCELLANY #26
By Earl Raguse

XB SPEED TESTS

I ran some experiments to find out which way to do things when more than method was available. I only got one surprise. In order to make my working of the stop watch less of a factor, I did each test a 1000 times, using a loop.

I remember that Jim Swedlow once wrote about this kind of thing in the ROM, and he reported how DEF functions were "glacially" slow. I have never used them or timed them, but I decided to test a few other things myself.

The first thing I did was to time a "do nothing" loop. That is LOOP1. Initially, I did not have "do nothing" lines 130 and 140 in the loop. It ran in 4.0 seconds. Then I thought it would yield more info, if the base loop included the two blank lines. That increased the run time by one second. I timed between BEEPS.

What does this tell us? I guess it says that it takes about .5 millisecond to find and to figure out that a line number is a do nothing line.

Henceforth, I will show only the lines that were made different than the previous loop. Thus LOOP2A just shows that line 130 DISPLAY AT(12,13)ERASE ALL:I to display the loop count was added. That increased the runtime to 1 Minute 46 Seconds.

LOOP2B changed 130 to CALL HCHAR(1,1, 32,768) to clear the screen, and 140 DISPLAY AT(12,13):I to display the loop count. Its runtime jumped to 15 M 10 S. Wow! not the way to do it.

Clearing the screen is expensive in computer time, no matter how you do it.

LOOP2C and LOOP2D experiment with partial screen clearing using a subprogram, CLS. LOOP2C's CLS uses DISPLAY AT in a loop, runtime 1 M 6 S, and LOOP2D uses CLS which in turn uses the loop produced by HCHAR(R,1,32,N) to clear the same four lines on the screen, in a runtime of 1 M 57.5 S. This apparently shows that DISPLAY AT is faster than HCHAR.

Later, in LOOP8 and 8A, I got a different result when I tried to clear only part of a line. Note that HCHAR normally works with a 32 character wide screen, and DISPLAY with a 28 character wide screen. Put on an equal basis, as i did in LOOP8 and 8A, it appears that HCHAR is faster, but not by much. That was my only surprise.

LOOP3 and 3A show the difference in doing a CALL CLEAR before displaying something repetitive. CALL CLEAR apparently takes .48 milliseconds to perform, ie 1000 of them takes 48 seconds.

LOOP4 shows that using a GOSUB instead of a direct call as in LOOP3A cost one 2 second per 1000, ie a GOSUB and RETURN takes about 2 milliseconds.

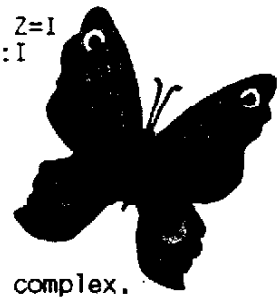
LOOP5 shows the same thing using a subprogram CALL DUB(I). Apparently using subprograms is slightly slower than subroutines. In this case 5.5 milliseconds longer, for a total of 7.5 milliseconds.

All methods of clearing the screen just write spaces there. So it boils down to which way is the fastest. That includes not doing unnecessary work.

In LOOP6 and 6A I tested the speed of IF logic vs what I call "direct" logic. There seems to be no significant difference. ie

```
130 IF I>100 THEN Z=I+2000 ELSE Z=I
140 DISPLAY AT(12,13) ERASE ALL:I
Runs in 1 M 25.8 S.
vs:
130 !
140 DISPLAY AT(12,13)ERASE ALL:
I-2000*(I>100)
```

Runs in 1 M 24.3 S.
I had expected a bigger margin.



LOOP7 and 7A get a little more complex. Here, IN 7, I am using a subprogram HPRT using HCHAR to both print the loop count and to clear the line. Runtime is 4 M 27 S. LOOP7A is the same, except no clearing. Runtime is 3 M 31 S. The display is actually improved. There is less flickering.

What this shows, is that DISPLAY AT (LOOP3A) is superior to HCHAR, but in console Basic, there was no choice, because DISPLAY is not in TI Console Basic vocabulary.

This also shows that one should not clear a line before printing messages of equal length. However, of course, if the message is shorter in length than the previous one, the shorter message will not completely over-write the prior longer message. Sorry, you can't always win, I guess.

TEST PROGRAM LISTING
By Earl Raguse

```
100 ! SAVE DSK1.LOOP1
110 DISPLAY BEEP
120 FOR I=1 TO 1000
130 !
140 !
150 NEXT I
160 DISPLAY BEEP
170 END
```

Runtime 0 M 5 S

```
100 ! SAVE DSK1.LOOP2A
130 DISPLAY AT(12,13)ERASE ALL:I
```

Runtime 1 M 46 S

```
100 ! SAVE DSK1.LOOP2B
130 CALL HCHAR(1,1,32,768)
140 DISPLAY AT(12,13):I
```

Runtime 15 M 10 S

```
100 ! SAVE DSK1.LOOP2C
130 CALL CLS(10,14)
140 DISPLAY AT(12,13):I

4300 SUB CLS(R1,R2):: FOR R=
R1 TO R2:: DISPLAY AT(R,1)
SIZE(30):: NEXT R :: SUBEND
```

Runtime 1 M 6 S

```
100 ! SAVE DSK1.LOOP2D
130 CALL CLS(10,4)
140 DISPLAY AT(12,13):I

4300 SUB CLS(R,N):: CALL
HCHAR(R,1,32,32*N):: SUBEND
```

Runtime 1 M 57.5 S

```
100 ! SAVE DSK1.LOOP3
130 CALL CLEAR
140 DISPLAY AT(12,13):I
```

Runtime 1 M 48 S

```
100 ! SAVE DSK1.LOOP3A
130 !
140 DISPLAY AT(12,14):I
```

Runtime 1 M 0 S

```
100 ! SAVE DSK1.LOOP4
130 GOSUB 1000

1000 DISPLAY AT(12,13):I :: RETURN
```

Runtime 1 M 2 S

```
100 ! SAVE DSK1.LOOP5
130 CALL DUB(I)
1000 SUB DUB(I):: DISPLAY AT(12,13
:I :: SUBEND
```

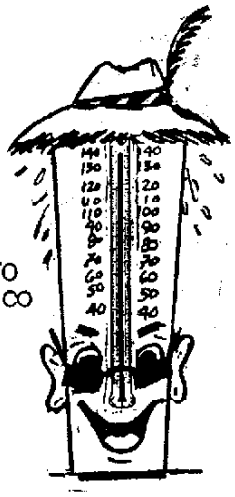
Runtime 1 M 7.5 S

```
100 ! SAVE DSK1.LOOP6
130 IF I>100 THEN Z=I+2000
ELSE Z=I
140 DISPLAY AT (12,13)ERASE ALL:Z
```

Runtime 1 M 25.6 S

```
100 ! SAVE DSK1.LOOP6A
130 !
140 DISPLAY AT(12,
13)ERASE ALL:I-2000*(I>100)
```

Runtime 1 M 24.3 S



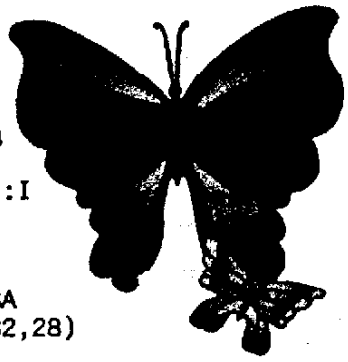
```
100 ! SAVE DSK1.LOOP7
105 CALL HCHAR(1,1,32,768)
125 B$=STR$(I)
130 CALL HPRT(B$,12,15)
140 CALL HCHAR(12,1,32,32)

6900 SUB HPRT(A$,ROW,COL)
6910 L=LEN(A$):: FOR I=1 TO
LEN(A$):: CALL HCHAR(ROW, CO
L+I,ASC(SEG$(A$,I,1))):
NEXT I
6920 SUBEND
```

Runtime 4 M 27 S

```
100 ! SAVE DSK1.LOOP7A
105 CALL HCHAR(1,1,32,768)
130 CALL HPRT(B$,12,15)
140 !

6900 SUB HPRT(A$,ROW,COL)
6910 L=LEN(A$):: FOR I=1 TO
LEN(A$)::CALL HCHAR(ROW, CO
L+I,ASC(SEG$(A$, I,1))):
NEXT I
6920 SUBEND
```



Runtime 3 M 31 S

```
100 ! SAVE DSK1.LOOP8
130 DISPLAY AT(12,1):
140 DISPLAY AT(12,13):I
```

Runtime 5 M 8 S

```
100 ! SAVE DSK1.LOOP8A
130 CALL HCHAR(12,3,32,28)
```

Runtime 4 M 58 S

XB MISCELLANY #27
By Earl Raguse

I hope there were at least some of you who actually tried to enter LISTMAN. I thought it was adequate for people who were trully interested in learning to program a little bit.

For the six year olds, and other senior citizens who can not follow what I have done thus far, I will recomend you stop by the Library, and pick up copies of the PUG Peripheral. Sue Harper has been running a programming series for children ages 2-7, (my estimate, not hers) which I deem to be excellent for any persons who had a great deal of difficulty in learning to read.

She spends and entire article telling people how to to use PRINT, and another on how to use CALL CLEAR. The most recent article in the April 1993 issue has advanced to FOR - NEXT to make a loop using CALL CLEAR and PRINT. She even goes so far as to nest FOR - NEXT loops to produce a delay.

Sue is a most patient teacher, and anybody who can not learn from her, is not really trying. That is not the end of Sue's talents, she writes articles for the rest of us too. I read her all the time.

It is not my intention, to do what Sue is doing, my goal is to teach reasonably inteligent persons, who are truly interested in learning, to program, and who willing to put in a little effort.

I will be most willing to do private tutoring for anyone who is having difficulty with a particular set of commands. All you have to do is talk to me at a meeting, or call me, 714/847-5875, and we can set up a mutually acceptable time. If necessary, to start with, I will take the Sue Harper approach, provided that you convince me that you want to learn, but can not seem to hack it for some reason. If you progress, then I will do it forever. I really do enjoy seeing someone learn about computing.

So much for digression. This month I wish to go into more detail on subprograms. Last month, I hope I didn't discourage anyone by showing that they were as much as two miliseconds slower than a subroutine. That two miliseconds is more than offset by all the other advantages of subprograms, not the least of which is your time.

One major advantage of subprograms is the fact that they keep their own set of variables. That is to say, that you my assign the name ZIGGY to a variable in your main program, and then safely use that name ZIGGY again within a subprogram, with no effect on ZIGGY in the main program. I will not discuss, how this is done, because I am not that conversant with the operation of XB. I am sure it has to do with assigning separate memory locations for variables.

Another advantage of subprograms is that they are called by name, and when they are MERGEd into a program, one does not need to know their location or line number to call them. The name, if properly chosen, tells the user what function the subprogram does, something not possible with subroutines.

Subprograms are not entirely divorced from the main program, however, they can both READ the same DATA statements.

There is no free lunch, however, because the usual purpose of a subprogram is to modify some variable according to your plan, and bring said results back to the program. In order to do that, the subprogram and its calling statement must include variable names, termed parameters, to pass the required info, into and out of the subprogram, ie CALL ZIGGY(A$(),X,Y). The subprogram does not have to have those exact parameter names in its definition, but they must be of the same type and same order.

At times it may be required to use a variable to pass infomation into a subprogram, but one does not want the subprogram to modify the variable in the main program. This can be easily accomplished by enclosing that one parameter X in parentheses, as follows:

```
155 CALL ZIGGY(A$(),(X),Y)
```

This CALL provides the subprogram with the information in X, but prevents the subprogram from changing this X even if the subprogram does modify its X.

Recall that these are entirely separate entities. Note that subscripted variables, like A$( ) do not contain any actual subscripts just the parentheses. All parameters used in the subprogram name must be in the CALL.

(cont next page)

One uses a subroutine when a certain sequence of instructions is repeated in a program, in order to save program space. One can use a subprogram even though it is used only once in a program, not to save space, but to save the programmer's time, and to make the program easier to read and understand. Once a subprogram is written, it may be easily MERGE'd into any other program.

Lets take a specific example, the thing that the TI XB Manual does little of. Lets use my most commonly used subprogram, PAK, means "Press Any Key To Proceed". The subprogram puts the above prompt sentence, on line 24, then awaits a press of any key before proceeding. It requires no parameters to be passed, nor does it return any.

A subprogram requiring action by the user, is, in my opinion, a better way than programming a waiting loop while the user reads a message or views an image. Some readers are a little slower, hence the wait must be long enough for the slowest reader. Also once having read the message, on previous uses of the program, one does not want to wait to proceed. Pressing any key just signals the program to get on with it.

```
6000 SUB PAK
6110 DISPLAY AT(24,1)SIZE(30)
:" Press Any Key to Proceed"
6120 CALL KEY(0,K,S) :: IF S<1
THEN 6120
6130 DISPLAY AT(24,1)SIZE(30)
:: SUBEND
```

Line 6000 is common to all subprograms, the subprogram must be named. line 6110 displays the prompt on screen row 24. line 6120 checks for a keypress, and not finding one returns to 6120 to look again. If one is found, ie S=1, line 6130 is executed to clear screen row 24, and the subprogram is exited by the SUBEND which returns to just after the to the calling statement.

All subprograms must have a SUBEND. An early exit can be made with SUBEXIT if desired. IF commands are not allowed in either of these exit lines.

The following subprogram GKEY expects two parameters in the calling statement, ie Q and ROW, where Q will be the ASCII value of the keypress, and ROW will be the screen row to display the keypress. It may be called as follows: CALL GKEY(Z,Y). The keypress will be returned in Z. Y is either a number, or must have been assigned a numbered row value. Note how Z is equated to Q and Y is equated to ROW, because of their order in the calling statement and the subprogram definition.

```
4000 SUB GKEY(Q,ROW)
4010 CALL KEY(3,K,S)::
IF S<0 THEN 4010 ELSE
Q=K :: IF Q=32 THEN A$=
"Space" ELSE A$=CHR$(Q)
4020 DISPLAY AT(ROW,1)
SIZE(30):" You Selected":
A$ :: SUBEND
```

This is very similar to PAK, but note that in line 4010, the value of K, (the ASCII value of the key pressed) is inserted into Q, and variable A$ is given the string value of Q. If the SPACE key was pressed, ie ASCII 32, then A$ is assigned "Space". Note that in line 4020 the actual keypress is displayed on ROW after the words "You Selected", then the subprogram is exited.

Note also the SIZE(30), after the DISPLAY AT statement, this clears the entire line before printing the selection result, SUBEND exits the subprogram. Once back at the calling line, a test can be made on Z to see what was actually pressed. For instance if the expected reply is selected from Y/N then the statement:

```
195 IF CHR$(Z)="Y" THEN xxx
ELSE if CHR$ (Z)="N" THEN yyy
ELSE zzz.
```

Where xxx and yyy are line numbers that execute the appropriate action, and zzz is the line number of the CALLing statement. One feature, not shown, is the first line in all my subprograms. For GKEY it is:

```
1 ! SAVE DSK1.GKEY,MERGE
```

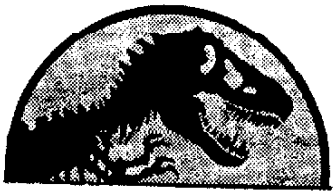
I have explained the use of that line in XB MISCELLANY #19.

To MERGE a subprogram, just say:

```
MERGE "DSK1.GKEY"
```

Check XB MISCELLANY #18 and #19 for an expalnation of these two statements.

Until next month, Happy Programming.



TIPS FROM THE TIGERCUB

No. 69

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213
$$$$$$$$$

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00 each. I am out of printed documentation so it will be supplied on disk.

My TI-PD library now has almost 600 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 and the latest supplement is available for $1 which is deductible from the first order.

In Tips #68 I published my solution to Dr. Ecker's challenge to alternately assign X the value of A and B without using IF...THEN or any outside help. Computer Monthly has arrived again and his solution is better than mine. Try it with any two numbers -

```
100 A=2.765 :: B=-10
110 X=A+B-X :: PRINT X :: GO
TO 110
```

There has been controversy for years as to whether the TI's psuedorandom number generator is truly random. Dr. Ecker's "Computer Fun & Learning" column in Computer Monthly had a question - if you randomly generate numbers between 0 and 9, how often will you get the same number twice in succession? Three times in succession? And etc. Since there are 10 numbers to choose from, it seems to me you would get 2 in a row 10% of the time, 3 in a row 1% of the time, 4 in a row .1%...etc. I wrote this to prove it -

```
100 RANDOMIZE
110 C=C+1 :: X=INT(RND*10)::
 PRINT X;:: IF X=F THEN FL=F
L+1 :: CL(FL)=CL(FL)+1 :: PR
INT "";FL;"=";CL(FL);"C=";C;
"%=";CL(FL)/C :: GOTO 110 EL
SE FL=0 :: F=X :: GOTO 110
```

After 10,000 tries, I had 2 in a row 8.75% of the time and 3 in a row .83% and 4 in a row .07% . Does that prove anything? I don't know.

(Dr. Ecker points out that those percentages could not ever quite add up to 100%!)

Here is another of my XBasic programs to write assembly source code -

```
100 DISPLAY AT(2,1)ERASE ALL
:"ASSEMBLY HELP SCREEN WRITE
R":"":" This program will wr
ite the":"source code for an
 assembly":"routine which ca
n be linked"
110 DISPLAY AT(7,1):"from Ex
tended Basic to dis-":"play
any one of several help":"sc
reens at any designated":"ke
y press or input at any":"po
int in a program."
120 DISPLAY AT(12,1):" The o
riginal source code,":"autho
r unknown, was improved":"by
 Karl Romstedt and further":
"modified by Bruce Harrison.
"
130 DISPLAY AT(20,1):"How ma
ny help screens?" :: ACCEPT
AT(20,24)SIZE(1)VALIDATE(DIG
IT)BEEP:N
140 FOR J=1 TO N :: H$=H$&"H
ELP"&STR$(J)&"," :: NEXT J ::
 H$="      DEF   "&SEG$(H$,
1,LEN(H$)-1)
150 DATA VMBW   EQU  >2024,V
MBR   EQU  >202C,KSCAN  EQU
 >201C,STATUS EQU  >837C
160 OPEN #1:"DSK1.HELP/S",OU
TPUT :: PRINT #1:H$ :: FOR J
=1 TO 4 :: READ M$ :: PRINT
#1:M$ :: NEXT J
170 FOR J=1 TO N :: H$="HELP
"&STR$(J):: PRINT #1:H$&"  L
WPI WS":"       LI   R13,HEL
PS"&STR$(J)
180 IF J<N THEN PRINT #1:"
     JMP  SAVSCR"
190 NEXT J :: H$=RPT$(" ",7)
200 PRINT #1:"SAVSCR CLR  R0
":H$&"LI   R1,SAVIT":H$&"LI
  R2,768":H$&"BLWP @VMBR":H$
&"LI   R9,NEWSCR":H$&"MOV  R
9,R1":H$&"MOV  R2,R4"
210 PRINT #1:H$&"LI   R3,>60
00":"ADDOFF MOVB *R13+,*R9":
H$&"AB   R3,*R9+":H$&"DEC  R
4":H$&"JNE  ADDOFF":H$&"BLWP
 @VMBW"
220 PRINT #1:"KEYLOO BLWP @K
SCAN":H$&"BLWP @KSCAN":H$&"C
B   @ANYKEY,@STATUS":H$&"JNE
  KEYLOO"
230 PRINT #1:"REPL   LI   R1
,SAVIT":H$&"BLWP @VMBW":"RET
N   LWPI >83E0":H$&"B    @>6
A"
240 PRINT #1:"WS     BSS  32
":"SAVIT  BSS  768":"NEWSCR
BSS  768":"ANYKEY BYTE >20":
H$&"EVEN"
250 DISPLAY AT(3,1)ERASE ALL
:" Enter data just as you":"
want it to appear, in 24":"l
ines. Press Enter for blank"
:"lines."
260 FOR J=1 TO N :: DISPLAY
AT(12,1):"Ready for screen #
"&STR$(J):"":"Press any key"
270 CALL KEY(0,K,S):: IF S=0
 THEN 270 ELSE CALL CLEAR
280 ACCEPT AT(1,0):M$ :: PRI
NT #1:"HELPS"&STR$(J)&" TEXT
 ' "&M$&RPT$(" ",30-LEN(M$))
&" '"
290 FOR K=2 TO 24 :: ACCEPT
AT(K,0):M$ :: PRINT #1:H$&"T
EXT ' "&M$&RPT$(" ",30-LEN(M
$))&" '"
300 NEXT K :: NEXT J :: PRIN
T #1:H$&"END"
310 DISPLAY AT(3,1)ERASE ALL
:" Source code has been writ
-":"ten to DSK1 as HELP/S. T
o":"assemble, insert Editor/
":"Assembler module."
320 DISPLAY AT(7,1):"Insert
Assembler disk in   drive 1
.":"Select 2 ASSEMBLER":"Loa
d Assembler? Y":"Source file
name DSK2.HELP/S"
330 DISPLAY AT(12,1):"Object
 file name? DSK2.HELP/O":"Li
st file name? Press Enter":"
Options? R"
340 DISPLAY AT(15,1):"Load t
he resulting object":"file i
nto your program by":"CALL I
NIT ::":"CALL LOAD(""DSK1.HE
LP/O"") or,"
350 DISPLAY AT(19,1):"much b
etter, iabed it with":"ALSAV
E or SYSTEX."
360 DISPLAY AT(21,1):"Access
 the screens in your progra
m by":" CALL LINK(""HELP1"")
":"CALL LINK(""HELP2""), etc
."
370 CALL KEY(0,K,S):: IF S=0
 THEN 370 ELSE CALL CLEAR
```

For instance, at any point in a program where keyboard input is required and user may not know what to do -

```
ACCEPT AT(24,1):M$ :: IF M$=
"HELP" THEN CALL LINK("HELP1
") and the first help screen
will pop up to give instruc-
tions. Press any key and the
previous screen reappears.
```

This time I am borrowing heavily from the TIIMES news letter of England, which has also borrowed from the REC newsletter.

This one is useless, but is a remarkable example of compact complex programming. It shows that there is an algorithm for everything. See if you can figure out how it works -

```
100 CALL CLEAR :: FOR A=1 TO
 2 :: FOR B=1 TO 4 :: X=2-AB
S(SGN(B-3)):: FOR C=1 TO X ::
 PRINT CHR$(64-7*A+5*B-8*X)
;:: NEXT C :: NEXT B :: PRIN
T CHR$(A+31):: NEXT A
```

Another useless one that is easier to figure out -

```
100 DISPLAY AT(1,1)ERASE ALL
:"NUMBER OF MONTH(1-12)"
110 ACCEPT AT(2,12)SIZE(2)VA
LIDATE(DIGIT):A :: IF A<1 OR
 A>12 THEN 110
120 DISPLAY AT(3,1):A;"x 4="
```

NEXT PAGE

```
;A#4 :: A=A#4
130 DISPLAY AT(4,1):A;"+13=";A+13 :: A=A+13
140 DISPLAY AT(5,1):A;"x 25=";A#25 :: A=A#25
150 DISPLAY AT(6,1):A;"-200=";A-200 :: A=A-200
160 DISPLAY AT(8,1):"Input date (1-31):" :: ACCEPT AT(8,19)SIZE(2)VALIDATE(DIGIT):B :: IF B<1 OR B>31 THEN 160
170 DISPLAY AT(10,1):A;"+";B;"=";A+B :: A=A+B
180 DISPLAY AT(11,1):A;"x 2=";A#2 :: A=A#2
190 DISPLAY AT(12,1):A;"-40=";A-40 :: A=A-40
200 DISPLAY AT(13,1):A;"x 50=";A#50 :: A=A#50
210 DISPLAY AT(15,1):"Input last two digits of    year e g 91:"
220 ACCEPT AT(16,16)SIZE(2)VALIDATE(DIGIT):B
230 DISPLAY AT(18,1):A;"+";B;"=";A+B :: A=A+B
240 DISPLAY AT(19,1):A;"-10500=";A-10500 :: A=A-10500
250 DISPLAY AT(24,1):"ANY KEY FOR ANOTHER"
260 CALL KEY(3,A,B)
270 IF B<1 THEN 260
280 RUN
290 END
```

One for the little ones - change the string to anything you want.

```
1 REM SILLY PROG BY S SHAW
  MARCH 1991
2 ! did you see COMPUTER WARS-the file? It is said that the star, who was required to type fast into a computer
3 ! could not type, so a program just like this one was used to give a good effect!
4 ! now adjust it how you wish and show your friends how fast you can type
5 ! at end of text string program will just stop with this listing but can be modified to do anything you wish!
6 !
100 A$="This is how a non-typist canproduce information on      screen quickly,without      "
110 A$=A$&"having to look at what keys are being bashed! Just bash keys and watch how perfect text appears no matter what you press."
120 CALL CLEAR :: PRINT A$;;;;;;
130 CALL KEY(3,A,B):: IF B<1 THEN 130
140 C=C+1 :: PRINT SEG$(A$,C,1);:: IF C=LEN(A$)THEN 160
150 GOTO 130
160 GOTO 160
```

And a very fast routine to find prime numbers -

```
100 ! FIRST 100 PRIMES
    -QUICKLY-
110 ! Dr H B Phillips
  from THE REC NEWSLETTER
 March 1988  Vol 3 #2
120 DIM P(300),X(12)
130 A=0 :: B=1 :: D=0.5 :: E=180
140 M=100 :: L=3 :: F=0
150 ! increase M for more- also increase DIMs.
160 PRINT 2;:: C=B :: IF M=B THEN END
170 L=INT((M/C)#L+F):: N=L+L+B
180 FOR I=B TO INT((SQR(N)-B)#D):: PP=P(I)
190 IF PP=B THEN 230
200 IF PP=A THEN PP=I+I+B :: PRINT PP;:: P(I)=PP :: C=C+B :: IF C=M THEN END
210 IF X(I)=A THEN X(I)=(PP#PP-B)#D
220 FOR J=X(I)TO L STEP PP :: P(J)=B :: NEXT J :: X(I)=J
230 NEXT I :: IF F=0 THEN S=I
240 FOR I=S TO L
250 IF P(I)=A THEN PP=I+I+B :: PRINT PP;:: P(I)=PP :: C=C+B :: IF C=M THEN END
260 NEXT I :: F=(M-C)#L/E :: S=L+B
270 GOTO 170
```

And a demonstration of how the INTERRUPT routine works independently of whatever else the computer is doing -

```
100 REM interrupt demo
110 REM
120 REM MACHINE LANGUAGE
130 REM ROUTINE LOADED AT
140 REM >2600 XB OR E/A WITH 32K
150 REM >7200 MINI MEM NO 32K
160 REM
170 CALL INIT
180 XM=9728
190 MM=29184
200 LAD=XM
210 REM TEST XB OR MM?
220 CALL LOAD(XM,170)
230 CALL PEEK(XM,X)
240 IF X=170 THEN 270
250 REM NO 32K MUST BE MM
260 LAD=MM
270 A=LAD
280 REM LOAD M/C
290 CALL CLEAR
300 FOR D=540 TO 630 STEP 10
310 CHECK=0
320 FOR N=1 TO 10
330 READ X
340 CALL LOAD(A,X)
350 CHECK=CHECK+X
360 A=A+1
370 NEXT N
380 READ X
390 IF CHECK<>X THEN 490
400 NEXT D
410 REM POKE INTERRUPT
420 REM ROUTINE ADDRESS
430 REM INTO >83C4
440 CALL LOAD(-31804,LAD/256)
450 REM JUST IDLE AWAY TIME
460 FOR N=1 TO 9940
470 NEXT N
480 STOP
490 PRINT "ERROR IN DATA STATEMENT ";D
500 STOP
510 REM EACH DATA STATEMENT
520 REM HAS 10 DATA BYTES
530 REM PLUS A CHECK SUM
540 DATA 192,236,000,092,004,194,005,131,002,131,987
550 DATA 000,060,026,003,004,195,006,236,000,094,624
560 DATA 203,003,000,092,060,172,000,090,006,002,628
570 DATA 017,015,019,010,006,002,019,004,002,000,94
580 DATA 002,039,010,083,016,002,002,000,002,086,242
590 DATA 096,003,016,007,002,000,000,119,010,083,336
600 DATA 016,002,002,000,000,072,160,003,002,096,353
610 DATA 064,000,006,192,215,192,006,192,215,192,1274
620 DATA 016,000,216,044,000,094,140,000,004,091,605
630 DATA 000,015,000,000,138,128,000,000,000,000,281
640 END
```

Run that, then press FCTN 4. Enter LIST. Enter NEW. To stop it, enter BYE.

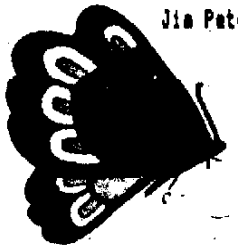
This is an oldie, but well worth repeating. You can use it to turn your cassette recorder on and off, to add speech or music from tape to a running program. With the proper hardware, you could write a program to control almost anything from the cassette port. If it doesn't work, reverse the polarity of the remote. Ed Hall wrote this -

```
100 CALL INIT
110 CALL LOAD(16368,79,70,70,32,32,32,36,252)
120 CALL LOAD(16376,79,78,32,32,32,32,36,244)
130 CALL LOAD(8194,37,4,63,240)
140 CALL LOAD(9460,2,12,0,45,29,0,4,91,2,12,0,45,30,0,4,91,203,78)
150 PRINT "PRESS":" P Play":"S Stop"
160 CALL KEY(3,A,B)
170 IF B<1 THEN 160
180 ON POS("PS",CHR$(A),1)+1 GOTO 160,190,200)
190 CALL LINK("ON"):: GOTO 160
200 CALL LINK("OFF"):: GOTO 160
```

And that is just about -

MEMORY FULL!

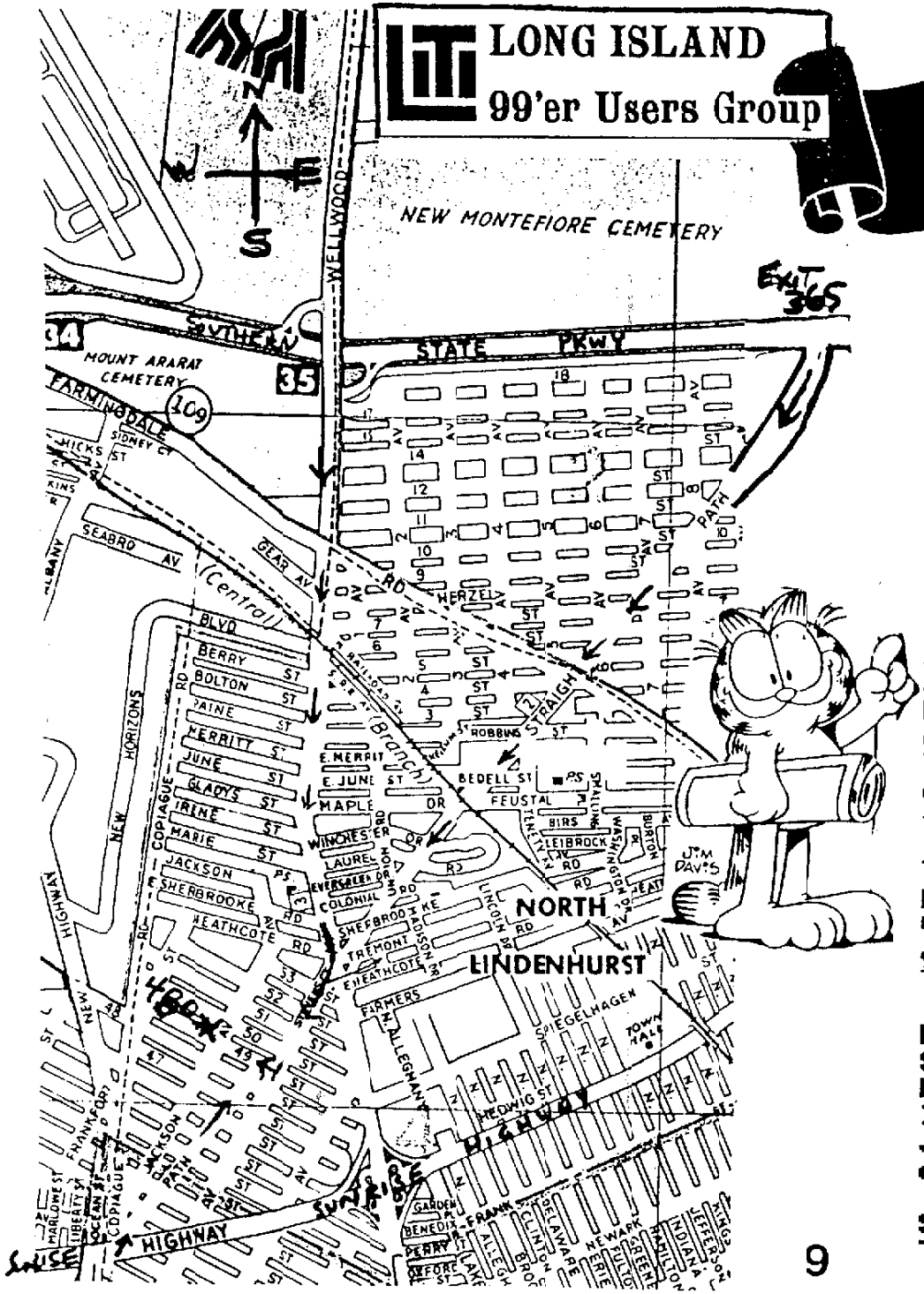
Jim Peterson



LONG ISLAND
99'er Users Group

NEW LOCATION

Meeting phone after
6 PM (516) 226-2529



NEW MEETING LOCATION (9/93)

Our GENERAL Meetings will be held at Tony's home in North Lindenhurst, NY. On the second friday of the month.

The address is: 480 Jackson Ave at 49th street.

Take SOUTHERN STATE:
Exit 35 South- WELLWOOD AVE. Go south on Wellwood. Make a right turn on Straight Path (Wellwood Commons Shopping Center). Go 5 blocks south to 49th street. Make a right turn on 49th, go one block to Jackson Ave. The house is on the corner with a fence, lantern post in the front yard.

Take SOUTHERN STATE:
Exit 36 South- STRAIGHT PATH to 49th street. Make a right turn, go one block to Jackson Ave. The house is on the NE corner.

From SUNRISE HIGHWAY:
STRAIGHT PATH NORTH for about 9 blocks to 49th street. Make a left turn, go one block to Jackson Ave. House is on NE corner.

See you at the meeting.
7:30 PM SHARP.....

Long Island

Long on Inspiration

VCR remote takes verbal orders

Canoga Park, CA—Now consumers can operate and program their VCRs with only a voice command. To record a show, users simply say, "Two, Monday, Nine p.m., Nine-thirty p.m."

The VCR Voice Programmer—a hand-held, battery-powered remote device from Voice Powered Technology Inc., Canoga Park, CA—also takes verbal orders. When, for example, a consumer wants to view the taped program, they say "play," or "pause," or "stop," "fast forward," and "rewind." Users can even use their voice to change channels on the television or cable box.

"The VCR Voice Programmer shows that easy-to-use, easy-to-understand, affordable voice recognition technology is not only possible but can solve a problem common to all of us," says Michael Bissonnette, founder and president of Voice Powered Technology.

The $169 VCR Voice Programmer features proprietary VoiceLogic voice recognition hardware and software housed in an ergonomic design to respond to the commands of up to four individual voices.

In a step-by-step learning process, the unit prompts each user to say a series of words and numbers via a built-in liquid crystal display. The device hears the words and memorizes them, and then is ready to respond to voice commands.

The unit programs up to 15 separate events, regardless of former capability of the VCR. The events can be next day, next week, or even years in advance. A single programmer can control two video systems.

For those viewers who choose not to view commercials, the special "zap it" feature in the device allows users to skip by them during playback. By saying "zap it" to the programmer, the VCR quickly advances the tape past commercials.

The VCR VOICE Programmer is the first of a line of VoiceLogic-controlled products being developed in the U.S. by Voice Powered Technology. The firm is dedicated to applying affordable voice recognition technology to control a range of consumer electronics, home appliances, office and automotive products. The Programmer will be exported to Europe and Japan following its U.S. introduction.

"Any consumer electronics device that is now controlled in some way by buttons or keypad can be controlled with the power of the voice," says Bissonnette. "Our aim is to bring about a fundamental change in people's ability to control a wide range of products and systems. To speed up that process, we are actively seeking licensees of our VoiceLogic technology." □

Remote programmer *uses voice recognition technology to take verbal orders from up to four individuals to program VCRs or operate televisions.*

LONG ISLAND SOUND
The Newsletter of the
LONG ISLAND USER'S U. G.

EDITOR: FRANK BUBENIK,JR

TONY IANNACONE.PRESIDENT
JERRY STOCKLER.VICE PRES
FRANK BUBENIK..SECRETARY
BOB LAWSON.....TREASURER

MEETINGS HELD ON SECOND
FRIDAY EACH MONTH AT
TONY'S HOME
840 JACKSON AVE.
NORTH LINDENHURST,NY

PHONE: (516) 938-1095
BBS NO:(516) 661-3643

>>> MAILING ADDRESS <<<
93 MYERS AVE
HICKSVILLE, NY
11801-2424

FIRST CLASS

DALLAS TI GROUP LISA SHAFFER
P.O. BOX 29863
DALLAS TX
75229 USA
9307 <-LAST NL RECEIVED